UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

| APPLICATION NO. | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|---|---|---|---|---|
| 09/874,170 | 06/04/2001 | Vasanth Bala | 10003355-1 | 7644 |

7590 03/31/2008

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P.O. Box 272400
Fort Collins, CO 80527-2400

| EXAMINER | |
|---|---|
| PROCTOR, JASON SCOTT | |
| ART UNIT | PAPER NUMBER |
| 2123 | |
| MAIL DATE | DELIVERY MODE |
| 03/31/2008 | PAPER |

**Please find below and/or attached an Office communication concerning this application or proceeding.**

The time period for reply, if any, is set in the attached communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

**BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES**

Application Number: 09/874,170
Filing Date: June 04, 2001
Appellant(s): BALA ET AL.

John P. Wagner, Jr. (35,398)
For Appellant

**EXAMINER'S ANSWER**

This is in response to the appeal brief filed 11 February 2008 appealing from the Office action mailed 10 October 2007.

**(1) Real Party in Interest**

A statement identifying by name the real party in interest is contained in the brief.

**(2) Related Appeals and Interferences**

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

**(3) Status of Claims**

The statement of the status of claims contained in the brief is correct.

**(4) Status of Amendments After Final**

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

**(5) Summary of Claimed Subject Matter**

The summary of claimed subject matter contained in the brief is correct.

**(6) Grounds of Rejection to be Reviewed on Appeal**

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

**(7) Claims Appendix**

The copy of the appealed claims contained in the Appendix to the brief is correct.

**(8) Evidence Relied Upon**

| | | |
|---|---|---|
| 6,370,687 | Shimura | 4-2002 |

**(9) Grounds of Rejection**

The following ground(s) of rejection are applicable to the appealed claims:

Claims 2-24 are rejected under 35 U.S.C. § 103(a) as being unpatentable over US Patent No. 6,370,687 to Shimura and further in view of Official Notice.

Regarding claim 2, Shimura teaches:

A networked system (column 2, lines 16-30) comprising:

a network (column 2, lines 16-30):

a server coupled to the network, wherein the server includes (column 2, lines 16-30):

an application code source that stores a client application, and a server code manager coupled to the application code source (Web server from which programs are retrieved, column 4, lines 27-35; Fig. 1, reference 20);

an application code transformation manager coupled to the application code source, for transforming the client application from a first format to a native binary format compatible with a native instruction set of the CPU of the client ("compile controller … recognizes the execute form of the client … and recognizes the native code of the execution environment in which the

compile unit 28 compiles from the byte code of the Java program"; column 4, line 52 – column 5, line 26); and

a server code segment manager coupled to the application code transformation manager, for parsing the client application in the native binary format into a plurality of code segments, said parsing of said code segments being dynamically performed based on actual server-side and client-side execution overhead, network bandwidth efficiency and client-side storage requirements on a per client basis, and configured based on predicted code segment usage or prior code segment usage history, wherein at least one of said plurality of code segments being transmitted to the client via the network [*"Then, in response to a request for the Java program from a client to the Web server on the network, the substitute compile server 10 returns to the client the Java program which has been* ***compiled and optimized into a native code conforming to the execute form of the requester client.****"* (column 6, lines 25-48, emphasis added)]; and

a client coupled to the network said client not having said client application stored thereon, wherein the client comprises (column 2, lines 16-30):

a CPU for natively executing at least one of said plurality of said code segments derived from the client application stored on said server (column 4, line 52 – column 5, line 26);

a code cache coupled to the CPU for storing said code segments (Official notice is taken that it is well known in the art to provide a CPU with a code cache. This Official Notice was first taken in the office action of 16 May 2005. Applicants have not traversed this use of Official Notice and as such, it is regarded as admitted prior art. See MPEP 2144.03 (C)); and

a client code manager-coupled to the code cache, for launching the client application by requesting that the server code manager transmit at least one of the plurality of dynamically

tailored code segments to the client (column 5, lines 27-32), receiving at least one of the dynamically tailored code segment from the server (column 5, lines 58-61), storing the dynamically tailored code segment in the code cache (Official Notice has been taken above regarding a code cache), and executing at least one of the plurality of dynamically tailored code segments using the CPU until the executed dynamically tailored code segment attempts to pass control to a required code segment not stored in the code cache (column 7, line 42 – column 8, line 18), at which point control passes back to the client code manager to retrieve the required code segment from the server, with the CPU continuing execution with the required code segment [*"...if the client side makes a request for the class file 42-2* [a next required code segment] *with successful prediction, then the compiled Java program can be executed at a high speed in an immediate response to the program request from the client side sine that program has already been compiled and retained on the cache unit 12."* (column 7, line 42 – column 8, line 18)].

It would have been obvious to a person of ordinary skill in the art at the time of Applicants' invention to implement the system taught by Shimura using a code cache in conjunction with the list of processors expressly taught by Shimura [*"SPARCV8, X86, Intel 486, etc."* (column 4, line 52 – column 5, line 26) because these processors either possess a code cache or it is well known in the art to use a code cache. The motivation for using a code cache is well known in the art, for example, to increase the effective speed of the processor by caching instruction code on the processor.

Regarding claim 3, Shimura teaches that the first format is other than the native execution format of the CPU of the client (column 5, lines 15-26). A compiler is functionally equivalent to a "transformation engine to transform the client application from the first format to the native binary format of the CPU of the client".

Regarding claim 4, Shimura does not explicitly teach that the first format is a source code text format of a programming language and the transformation manager comprises a compiler that compiles and links the client application into a native binary format of the CPU of the client. However, Shimura does explicitly teach that the first format is a "Java program in the form of the virtual machine computer program prepared as an applet on the web page" (column 4, lines 28-30) which is compiled using a Java™ compiler (column 4, lines 42-51 and throughout). It would have been obvious to a person of ordinary skill in the art that the term "Java applet" commonly refers to source code in a text format intended for use in a web page and that source code in a text format is well known input to a typical compiler. It therefore would have been obvious to a person of ordinary skill in the art to implement Shimura's system where the first format is a source code text format of a programming language and compiling that source code into a native binary format of the CPU of the client.

Regarding claim 5, Shimura teaches that the transformation manager comprises a just-in-time compiler (column 5, lines 8-15).

Regarding claim 6, Shimura's teaching regarding class files (column 7, line 42 – column 8, line 18) would be obvious to a person of ordinary skill in the art at the time of Applicants' invention as functionally equivalent to "code segments". It would be obvious to a person of ordinary skill in the art at the time of Applicants' invention to implement this functionality with a client code manager that requests needed segments from the server and to branch into the received code segment. Indeed, this is the functionality implied by Shimura (column 7, line 57 – column 8, line 13) although the obvious details of implementation are omitted.

Regarding claim 7, Shimura does not explicitly recite the steps of adjusting branch instructions to link into and out of received code segments as recited. Shimura implies this functionality (column 7, line 42 – column 8, line 18; column 9, line 63 – column 10, line 9). Official notice is taken that the need to link code that is compiled in sections is well known. It would have been obvious to a person of ordinary skill in the art at the time of Applicants' invention, in combination with his own knowledge of the particular art, to adequately support linking sections of compiled code by adjusting the branch instructions. Failure to do so would create an inoperable system, as would be recognized as well known by a person of ordinary skill in the art. Applicants have not traversed this use of Official Notice and as such, it is regarded as admitted prior art. See MPEP 2144.03 (C).

Claim 8 recites what is generally known in the art as "garbage collection". Official notice is taken that Java™ and the Java™ virtual machine support garbage collection. It would have been obvious to a person of ordinary skill in the art at the time of Applicants' invention, in

combination with his own knowledge of the particular art, to implement the system taught by Shimura using garbage collection because of the well-known advantages of garbage collection, such as ease of programming and recovery unused memory.

Claims 9-13 recite the server portion of the system of claims 1-5 and are rejected for the same reasons given above for claims 1-5.

Claims 14-17 recite the client portion of the system of claims 1 and 6-8 and are rejected for the same reasons given above for claims 1 and 6-8.

Claims 18-22 recite the methods performed by the system of claims 1-7 and are rejected for the same reasons given above for claims 1-7.

Claims 23-24 recite a computer program product and system according to claims 1-7 and are rejected for the same reasons given above for claim 1.

**(10) Response to Argument**

In response to the Final Rejection shown above, Appellants argue primarily that: Regarding Independent Claims 2, 9, 14, 18, and 23, Appellants respectfully submit that Claim 2 (Claims 9, 14, 18, and 23 include similar features) includes the features "a server code segment manager coupled to the application code transformation manager, for parsing the client application in the native binary format into a plurality of code segments, said parsing of said code segments being dynamically performed based on actual server-side and client-side execution overhead, network bandwidth efficiency and client-side storage requirements on a per client basis, and configured based on predicted code segment usage or prior code segment usage history" (emphasis added by Appellants).

Appellants respectfully submit that Shimura does not teach this claimed feature. That is, Appellants do not understand Shimura to teach dynamic parsing of application code into code segments wherein the parsing of the code segments is dynamically performed based on actual server-side and client-side execution overhead, network bandwidth efficiency and client-side storage requirements on a per client basis (emphasis added by Appellants).

The Examiner respectfully traverses this argument as follows.

Shimura discloses dynamic parsing of application code [*"If in step S1 the compile server 10 accepts a request for access to a Java program of the Web server 20 from the client 14-1 for example, then in step S2 a check is made to see whether the requested Java program is retained in the cache unit 12.* [...] *If no compiled Java program lies on the cache unit 12...* [several intermediate steps are performed, culminating with] *"the compile controller 22 specifies one of the plurality of JIT compilers as shown in FIG. 5 corresponding to the execute form correspondent to the client name of the requesting client name of the requesting client 14-1 which has been recognized through the reference to the control memory, and it compiles the Java program in step S4. After having been translated and optimized on a method-to-method basis into a native code in the execute form of the requesting client 14-1 by the compile unit 28, the compiled Java program is cached in the cache unit 12. At the same time, the compiled Java*

*program is returned via the LAN interface 30 to the requesting client 14-1 in which the Java program comprised of the compiled native code is run."* (column 5, lines 27-61); *"Then, in response to a request for the Java program from a client to the Web server on the network, the substitute compile server 10 returns to the client the Java program which has been compiled and optimized into a native code conforming to the execute form of the requester client."* (column 6, lines 25-30)]. "Parsing" is a necessary and inherent step to compile application code such as Java code and there appears to be no argument that generic "parsing" is disclosed in Shimura.

To summarize the preceding, in Shimura, a client 14-1 makes a request for access to a Java program from the web server 20. When the web server does not find the pre-parsed, pre-compiled application in its cache 12, the web server initiates a process to dynamically parse the application (i.e. dynamically in that the parsing and compiling happens in real time and in response to a real time request) and provides the resulting dynamically parsed, dynamically compiled application code to the client.

Thus Shimura discloses dynamic parsing of application code within the meaning of the claim language. Further, Appellants' specification fails disclose a deliberate or special definition of "dynamic parsing" beyond what the Examiner has shown above. If, to Appellants, "dynamic parsing" means something other than parsing in real time and in response to a real time request, that definition is not found in the specification.

Shimura discloses parsing code segments based on actual server-side and client side execution overhead, network bandwidth efficiency, and client-side storage requirements on a per client basis. Appellants' specification provides no definition, explanation, or disclosure of these

features beyond reciting the words (For example, in the Specification at page 15, lines 2-4). Each will be addressed independently.

"Execution overhead" as understood by a person of ordinary skill in the art means the processing steps required to manage the execution of an application which are not themselves part of the application. For example, if executing an application on a client requires the additional overhead steps of requesting additional application code from a remote server, those steps are considered "execution overhead". If the client had the entire application, the steps of requesting and receiving the additional application code would not be performed. These steps are not part of the application code itself, but are required to manage the execution of an application, and these steps are not themselves part of the application. As shown above, Shimura discloses dynamic parsing based on actual server-side and client side execution overhead.

"Network bandwidth efficiency" as understood by a person of ordinary skill in the art is merely one of many different measures of network traffic. It is inherent that when there is "traffic" on a network, "network bandwidth" is used. When any network bandwidth is used, there exists some "network bandwidth efficiency". That efficiency may be good, bad, or mediocre, but it exists and may be measured. As shown above, Shimura plainly discloses network traffic by disclosing the transfer of requests and application code between a client and a server. Thus, Shimura discloses dynamic parsing based on actual server-side and client side network bandwidth efficiency.

"Client-side storage requirements" as understood by a person of ordinary skill in the art means the storage required for a client to properly operate. These requirements might be the storage space required to store an application program so that it can be properly executed.

Shimura clearly discloses a client that receives an application program and executes it, as shown above. Thus, Shimura teaches dynamic parsing based on actual client-side storage requirements.

As submitted by the Examiner in the final Office Action and quoted in Appellants' arguments, "[T]he breadth of the limitation hinges upon the phrase "based on" which is clearly expansive." (emphasis added) Appellants have neither disclosed nor specifically claimed any particular steps, features, or functionality to further limit or suggest an interpretation for the phrase "based on" as used in the claim language. The Examiner has shown above how Shimura performs dynamic parsing according to Appellants' specification and claims, and further how that dynamic parsing is based on actual server-side and client-side execution overhead, network bandwidth efficiency, and client-side storage requirements. The dynamic parsing is based on these features because proper operation of the Shimura invention involves dynamic parsing that requires execution overhead, requires network bandwidth efficiency, and respects client-side storage requirements.

Lastly, Appellants' brief contains a response to the Examiner's remarks in the final Office Action. The Examiner submits that Appellants' arguments at this point significantly mischaracterize the Examiner's intent, meaning, and conclusions. To the extent that the Examiner's previous remarks were unclear, the Examiner apologizes for the misunderstanding. The remarks above better explain the same analysis performed by the Examiner in the final Office Action.

As a result, the Examiner finds Appellants' argument that "not parsing is not analogous to dynamic parsing," et cetera, moot because the Examiner does not make those arguments.

The Examiner has fully considered Appellants' arguments, but for the reasons set forth above, Appellants' arguments have been found unpersuasive. Claims 2-24 are obvious of Shimura in view of Official Notice.

**(11) Related Proceeding(s) Appendix**

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

Jason Proctor

/Paul L Rodriguez/

Supervisory Patent Examiner, Art Unit 2123

Conferees:

Paul Rodriguez

/Paul L Rodriguez/

Supervisory Patent Examiner, Art Unit 2123

Eddie Lee

/Eddie Lee/

Supervisory Patent Examiner, TC 2100